ISSN 1870-4069

Lógica difusa y el manifiesto ágil: Innovación en la medición de agilidad en el desarrollo de software

Sergio Octavio Rosales Aguayo[1,3], Pedro Damián Reyes[2], José Román Herrera Morales[2], Ricardo Acosta Díaz[2]

[1] Tecnológico Nacional de México, Jalisco, México

[2] Universidad de Colima, México

[3] Universidad Davinci, México

sergio.ra@cdguzman.tecnm.mx, {damian, rherrera, acosta}@ucol.mx

Resumen. Las metodologías ágiles son esenciales en el desarrollo de software; sin embargo, cuantificar la agilidad sigue siendo un desafío. Este estudio presenta un enfoque que utiliza la lógica difusa, reconocida por manejar la imprecisión y ambigüedad, para medir la agilidad en el desarrollo de aplicaciones. Se establece una matriz de correspondencia y una distribución de frecuencias que informa un marco de medición de agilidad basado en lógica difusa. La metodología propuesta se detalla en etapas secuenciales y se representa esquemáticamente con diagramas de actividad UML. El sistema de inferencia difuso, creado con Python, integra funciones de membresía predefinidas y 81 reglas. Además, se desarrolló un instrumento de medición aplicado a equipos de desarrollo frontend que provee datos para el sistema difuso, resultando en métricas de agilidad cuantificables. La eficacia de esta metodología se demuestra en estudios de caso reales, mostrando una alineación con la agilidad percibida y proporcionando una métrica objetiva y matizada de agilidad. Este estudio sienta las bases para futuras exploraciones y el refinamiento continuo de prácticas ágiles, proponiendo el modelo basado en lógica difusa como un estándar potencial para evaluar la agilidad en proyectos de software.

Palabras clave: Lógica difusa, desarrollo ágil, evaluación de la agilidad en desarrollo de software.

Fuzzy Logic and the Agile Manifesto: Innovation in Measuring Agility in Software Development

Abstract. Agile methodologies are essential in software development; however, quantifying agility remains a challenge. This study presents an approach that uses fuzzy logic, known for handling imprecision and ambiguity, to measure agility in application development. A correspondence matrix and frequency distribution

are established that inform an agility measurement framework based on fuzzy logic. The proposed methodology is detailed in sequential stages and represented schematically with UML activity diagrams. The fuzzy inference system, created with Python, integrates predefined membership functions and 81 rules. Additionally, a measurement instrument was developed applied to frontend development teams that provides data for the fuzzy system, resulting in quantifiable agility metrics. The effectiveness of this methodology is demonstrated in real case studies, showing alignment with perceived agility and providing an objective and nuanced metric of agility. This study lays the foundation for future exploration and continuous refinement of agile practices, proposing the fuzzy logic-based model as a potential standard for evaluating agility in software projects.

Keywords: Fuzzy logic, agile development, agility evaluation in software development.

1. Introducción

En el dinámico mundo del desarrollo de software, la agilidad es más que una metodología; es una necesidad vital para la supervivencia y el éxito. El Manifiesto por el Desarrollo Ágil de Software [1], ha inspirado a equipos de todo el mundo a adoptar prácticas que promueven la adaptabilidad y la respuesta eficiente al cambio [2]. Sin embargo, medir la agilidad de una manera que capture su esencia multifacética sigue siendo un desafío [3], la lógica difusa, con su capacidad intrínseca para manejar la imprecisión y la ambigüedad, se postula como una solución prometedora. Este estudio presenta un modelo innovador que fusiona la lógica difusa con los principios ágiles y cuatro metodologías populares para medir la agilidad en el desarrollo de aplicaciones front end.

2. Revisión de la literatura

2.1. Desarrollo de software

El desarrollo de software implica la creación, diseño, despliegue y soporte de aplicaciones y sistemas software. Este proceso puede adoptar diversas metodologías que estructuran, planifican y controlan el desarrollo de un sistema de información. Entre los modelos tradicionales se encuentran:

- Modelo Cascada: donde el proceso sigue secuencias lineales y fases estancas, como requisitos, diseño, implementación, verificación y mantenimiento [4].
- Modelo Espiral: que combina elementos iterativos con evaluaciones de riesgo en cada ciclo [5].
- El proceso de Desarrollo de Software Unificado: un enfoque iterativo que enfatiza la adaptabilidad y la entrega de software operativo en cada iteración [6].

Los modelos anteriores fueron considerados pesados en términos de documentación extensa y tiempos largos de análisis y diseño. En respuesta a las limitaciones de estos

Fig. 1. Metodología de trabajo.

modelos tradicionales, surge el Manifiesto por el Desarrollo Ágil de Software [1], que propone valores y principios orientados a la adaptabilidad y la entrega continua. Las metodologías ágiles como Scrum, Kanban y Extreme Programming (XP) se enfocan en la colaboración, el cliente como centro y la capacidad de adaptarse a cambios rápidos [7-8]. Finalmente, la medición de la agilidad en el desarrollo de software se ha comenzado a abordar mediante técnicas avanzadas como la inteligencia artificial, el aprendizaje automático y la lógica difusa, ofreciendo métodos más sofisticados y adaptativos para evaluar y mejorar las prácticas de desarrollo ágil [9-10].

2.2. Metodologías ágiles comunes

Por otro lado, las metodologías ágiles se fundamentan en una serie de principios y valores que promueven un enfoque de desarrollo de software más flexible y orientado a las personas, Meyer [11] menciona que las metodologías de mayor uso son: Scrum desarrollada por Jeff Sutherlan y Ken Schwaber [7], Lean Software desarrollada por Mary Poppendieck [12], Crystal desarrollada por Alistair Cockburn [13] y Extreme Programming (XP) desarrollada por Kent Beck [14].

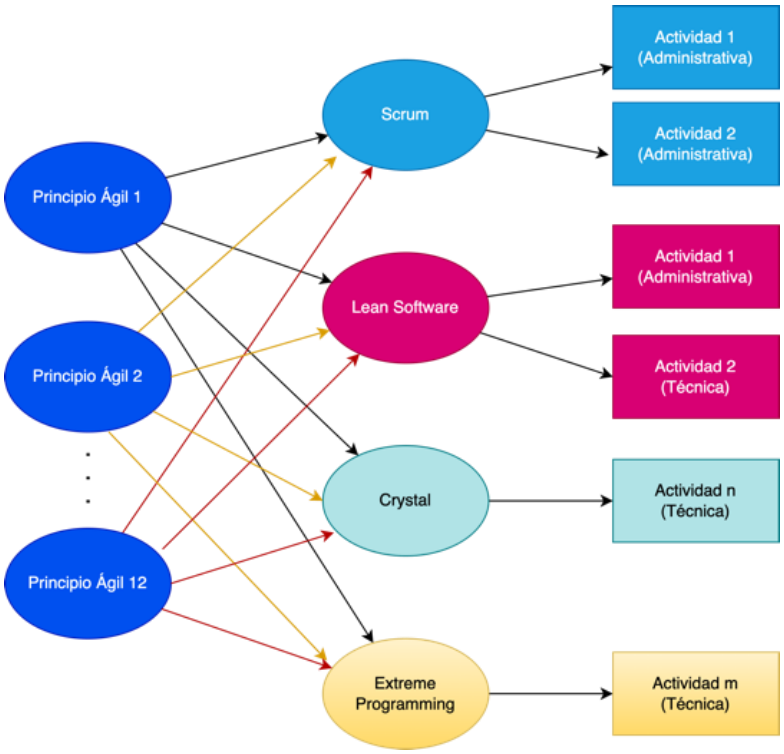

Fig. 1. Correspondencia entre principios ágiles, metodologías y actividades.

2.3. Lógica difusa

La lógica difusa es una rama de la Inteligencia Artificial que permite analizar información entre lo falso y lo verdadero [15]. El objetivo principal de la lógica difusa es crear un sistema basado en el comportamiento y pensamiento humano, con lo cual se pueden resolver problemas de ciencias actuariales, administración y gestión de empresas, química, ciencias de la tierra, ecología y ciencias ambientales, economía, ingeniería (civil, industrial, mecánica, nuclear, etc.), ergonomía, tecnología de la información, medicina, ciencias sociales, telecomunicaciones, gestión del tráfico [16]. La lógica difusa está basada en teoría de conjuntos difusos, la cual es una generalización de la teoría clásica de conjuntos [17].

2.4. Agilidad y lógica difusa

Los fundamentos de la agilidad y la lógica difusa tienen raíces profundas tanto en la teoría como en la práctica. El MDAS, ha sido un catalizador para la transformación de procesos de desarrollo de software.

La lógica difusa, propuesta por Lotfi Zadeh [18], ha evolucionado hasta convertirse en una herramienta poderosa para el modelado de decisiones en situaciones inciertas.

Tabla 1. Selección de principios, la actividad correspondiente y la metodología a la que pertenece.

Principio	Actividad	Metodología
Principio Ágil 1	Frequent Integration	Crystal Method
	Small Releases	Extreme Programming
Principio Ágil 2	Refactoring	Lean Software
	Frequent Delivery	Crystal Method
	TDD	Extreme Programming
Principio Ágil 3	Frequent Delivery	Crystal Method
	Small Releases	Extreme Programming
Principio Ágil 7	Frequent Delivery	Crystal Method
	Continuous Integration	Extreme Programming
Principio Ágil 9	Refactoring	Lean Software
	Refactoring	Extreme Programming
Principio Ágil 10	Simple Design	Extreme Programming

Tabla 2. Distribución de frecuencia de actividades ágiles de tipo técnico.

Actividad	Frecuencia	Porcentaje
Continuous Integration	2	15%
Frequent Delivery	5	38%
Refactoring	4	31%
Testing	1	8%
Simple Design	1	8%

Los trabajos de Beck y colaboradores [14] sobre prácticas de desarrollo ágil y de Ross sobre lógica difusa [19], permiten integrar ambas áreas del conocimiento.

3. Metodología

La metodología que se propone en este artículo se construyó a partir de una revisión exhaustiva de la literatura sobre el Manifiesto Ágil y cuatro metodologías ágiles prominentes. Se desarrolló una matriz de correspondencia para identificar y sintetizar los principios y metodologías que fundamentan la práctica ágil.

Posteriormente, se clasificaron las actividades de desarrollo en administrativas y técnicas para enfocar la evaluación de agilidad en aquellas actividades que directamente contribuyen a la entrega de valor.

Se elaboró una distribución de frecuencias para las actividades identificadas con el fin de determinar su relevancia en la práctica ágil actual. Con estos datos, se diseñó un marco de medición que emplea la lógica difusa para cuantificar la agilidad, permitiendo una evaluación más flexible y representativa de las prácticas ágiles en diferentes contextos. Esta metodología se representa de forma esquemática en la Fig. 1.

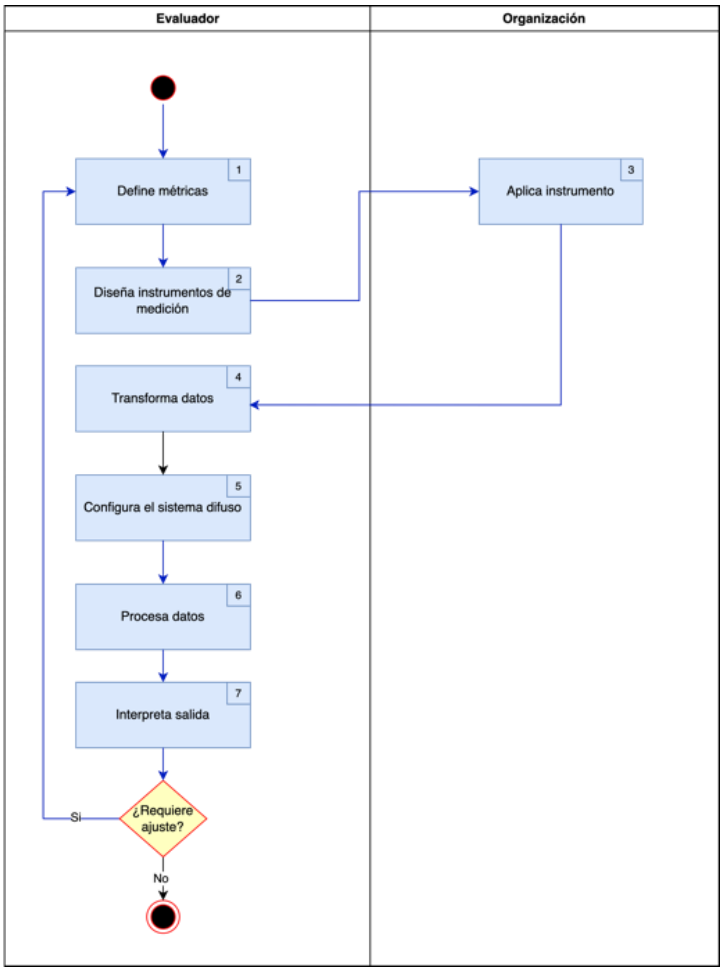

Fig. 3. Metodología para medir el nivel de agilidad de desarrollo.

La Fig. 2, muestra la relación entre los doce principios ágiles fundamentales, como se definen en el Manifiesto Ágil, y cuatro metodologías ágiles prominentes: Scrum, Lean Software, Crystal y Extreme Programming. Se distinguen dos categorías de actividades de desarrollo de software: administrativas y técnicas.

Las actividades administrativas son aquellas que se relacionan con la gestión del proyecto, mientras que las técnicas se refieren a las prácticas directamente involucradas en la creación del software. La Tabla 1 presenta una asociación detallada entre seis principios ágiles seleccionados, las actividades de desarrollo de software clave y las metodologías ágiles que típicamente las implementan.

Los datos se organizan en tres columnas: la primera lista los principios ágiles numerados; la segunda, las actividades de desarrollo de software relacionadas con ese principio; y la tercera, las metodologías ágiles que adoptan esas actividades. Esto proporciona un marco para comprender cómo las metodologías ágiles pueden ser

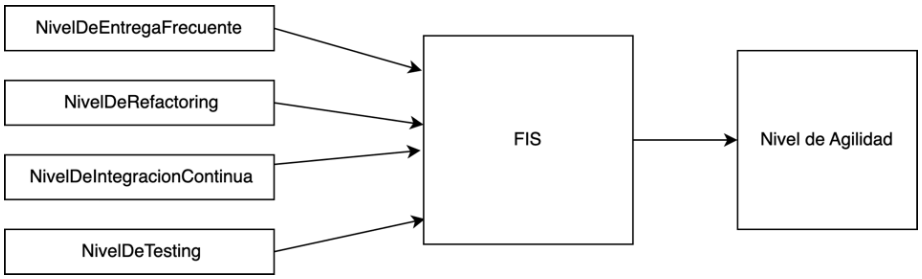

Fig. 2. Sistema de inferencia difuso con 4 variables de entrada.

Tabla 3. Variables de entrada.

Actividad	Variable	Relevancia	Términos lingüísticos
Frequent Delivery (Entrega Frecuente)	NivelDeEntregaFrecuente (NEF)	38%	Bajo (BNE)
			Medio (MNE)
			Alto (ANE)
Refactoring+Simple Design (Refactorización)	NivelDeRefactoring (NR)	39%	Bajo (BNR)
			Medio (MNR)
			Alto (ANR)
Continuous Integration (Integración Continua)	NivelDeIntegracionContinua (NIC)	15%	Bajo (BNIC)
			Medio (MNIC)
			Alto (ANIC)
Testing (Pruebas)	NivelDeTesting (NT)	8%	Bajo (BNT)
			Medio (MNT)
			Alto (ANT)

Tabla 4. Salida del Sistema Difuso.

Variable lingüística	Términos lingüísticos
Nivel de agilidad	Nivel de Agilidad BAJO (NAB)
	Nivel de Agilidad Medio (NAM)
	Nivel de Agilidad Alto (NAA)

aplicadas selectivamente para reforzar principios específicos en un entorno de desarrollo de software.

En referencia a la Tabla 1, se llevó a cabo un análisis para discernir y consolidar las actividades del desarrollo ágil.

Se observó la presencia de actividades recurrentes y otras que, pese a tener denominaciones distintas, compartían esencia y propósito. Esta evaluación culminó con la identificación de cinco actividades fundamentales que representan prácticas centrales en metodologías ágiles: 'Continuous Integration', 'Frequent Delivery', 'Refactoring', 'Testing', y 'Simple Design'. Estas actividades conforman la base para la posterior elaboración de la propuesta de la metodología de medición de agilidad.

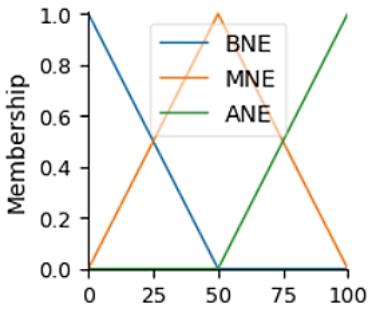

Fig. 5. Funciones de membresía para las variables de entrada.

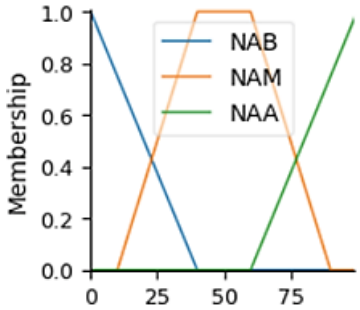

Fig. 6. Funciones de membresía de NivelDeAgilidad.

Tabla 5. Nivel de Agilidad Empresa de las empresas E001, E002, E003, E004, E005 y E006.

Variable Lingüística	**E001**	**E002**	**E003**	**E004**	**E005**	**E006**
Nivel de Entrega Frecuente	56	63	55	38	15	20
Nivel de Refactoring	34	65	65	46	11	28
Nivel de Integración Continua	80	73	60	60	16	18
Nivel de Testing	68	90	58	36	17	22
Nivel de Agilidad Calculado	**NAM (55.07)**	**NAM (57.07)**	**NAM (53,75)**	**NAM (49.07)**	**NAB (35.56)**	**NAB (39.83)**

La Tabla 2, muestra un análisis cuantitativo de la frecuencia con la que se implementan ciertas actividades esenciales dentro de las metodologías ágiles, según se ha derivado de la Tabla 1. Las actividades se enumeran en la columna izquierda, mientras que la columna central refleja el número de veces que cada actividad es referenciada o aplicada dentro del contexto de la investigación actual, la columna de la derecha representa el porcentaje en términos de relevancia.

Frequent Delivery encabeza la lista con un 38% en términos de relevancia, seguido de Refactoring con un 31%. A estos le siguen Continuous Integration con un 15% y Simple Design y Testing, ambos con un 8%. Por lo tanto, la conclusión que se extrae es que estas actividades deben ser consideradas como prioritarias en cualquier implementación de metodologías ágiles, dada su significativa influencia en la programación y en el alineamiento con los principios ágiles del MDAS.

4. Metodología de medición propuesta

Con estos antecedentes, se ha diseñado una metodología estructurada para la evaluación de la agilidad en el desarrollo de aplicaciones front end utilizando la lógica difusa. Esta metodología se describe a través de un Diagrama de Actividad de UML

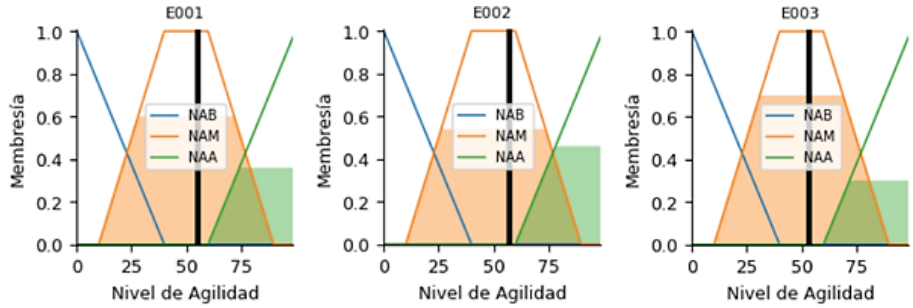

Fig. 3 Niveles de Agilidad de las empresas E001, E002 y E003.

representado en la Fig. 3. El diagrama ofrece una representación visual y sistemática del proceso de evaluación, mismo que se detalla a continuación.

1. **Define métricas**: Establecer claramente las métricas que definen, para este caso las métricas son: 'NivelDeEntregaContinua', 'NivelDeIntegraciónContinua', 'NivelDeRefactoring' y 'NivelDeTesting'.
2. **Diseña instrumentos de medición:** Se crearon cuatro instrumentos o herramientas para obtener valoraciones cualitativas (alto, medio, bajo) para cada métrica.
3. **Aplica instrumento:** Se aplican los instrumentos a equipos de desarrollo front end para recabar las valoraciones.
4. **Transforma datos:** Se traducen las valoraciones cualitativas a valores difusos utilizando funciones de membresía predefinidas.
5. **Configura el sistema difuso:** Se utiliza Python para configurar el sistema de inferencia de Mandani con las 81 reglas ya establecidas.
6. **Procesa datos:** Ingresar los valores difusos en el sistema de Mandani y realizar la inferencia difusa.
7. **Interpreta salida:** Decodifica la salida difusa del sistema para obtener un nivel de agilidad claro (bajo, medio, alto).
8. **Valida y Ajusta:** Contrasta los resultados con evaluaciones expertas de agilidad para validar la precisión del sistema difuso y realiza ajustes si es necesario.

Esta metodología provee una guía estructurada para la evaluación de la agilidad en proyectos de desarrollo front end, facilitando un enfoque sistemático y replicable.

5. Implementación del sistema difuso

La Tabla 2, muestra las actividades seleccionadas con su relevancia, mismas que fueron consideradas inicialmente para la construcción del sistema difuso. Sin embargo, con el fin de optimizar el sistema en mención, se propuso fusionar las actividades Refactoring y Simple Design. Beck y colaboradores [14], sostienen que la refactorización continua es un componente crítico en el sostenimiento de un diseño simple, proporcionando la flexibilidad necesaria para acomodar cambios en los

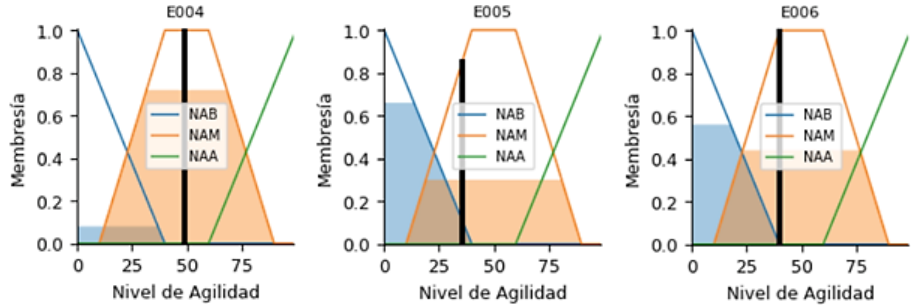

Fig. 4 Niveles de agilidad de las empresas E004, E005 y E006.

requisitos sin incrementar la complejidad del sistema. Para estas actividades, se definen las variables lingüísticas.

5.1. Definición de variables lingüísticas

Las variables lingüísticas son variables cuyos valores no son números sino términos o etiquetas lingüísticas en un lenguaje natural o artificial.

En la lógica difusa, permiten la manipulación de conceptos aproximados como "alto", "medio", o "bajo" y son utilizadas para describir tanto los antecedentes como los consecuentes de las reglas If-Then [20]. En este caso, las variables lingüísticas están asociadas con las actividades de agilidad en el desarrollo de aplicaciones. A cada actividad se le asocia una variable lingüística que describe el nivel de desempeño de la actividad correspondiente. El sistema está entonces definido por medio de 4 entradas y una salida conforme se muestra la Fig 4.

El esquema operativo presentado en la Fig. 4 incorpora un conjunto de reglas If-Then, fundamentales para la lógica difusa, que establecen una relación causal entre condiciones específicas (antecedentes) y resultados esperados (consecuentes). Mediante la aplicación de la inferencia difusa, el sistema traduce entradas con grados de verdad parciales en una salida difusa coherente, siguiendo un proceso de razonamiento aproximado que simula la capacidad humana de tomar decisiones en situaciones ambiguas o inciertas [19].

5.2. Definición de reglas If-Then

Para construir las reglas If-Then en un sistema de lógica difusa que maneja variables con tres grados de membresía, se emplea la Ecuación (1). Basándose en esta formulación, y utilizando el operador lógico 'y' (AND), se determina que un sistema que comprende 4 variables con tres grados de membresía cada una, se necesitan un total de 81 reglas If-Then. Este cálculo se detalla en la Ecuación (2):

$$\text{Núm. de reglas} = \text{Núm. de grados de membresía}^{\text{Núm. de variables}}, \tag{1}$$

$$\text{Núm de reglas} = 3^4 = 81. \tag{2}$$

Tabla 6 Años de Experiencia y Valor de Agilidad.

Empresa	Años de Experiencia	Valor De Agilidad	Nivel De Agilidad
E001	7	55.07	NAM
E002	15	57.08	NAM
E003	20	53.75	NAM
E004	7	49.07	NAM
E005	20	35.56	NAB
E006	4	39.83	NAB

Las variables lingüísticas con sus pesos y términos lingüísticos se definen como se muestra en la Tabla 3. De igual manera, la variable de salida denominada NivelDeAgilidad, tendrá los términos lingüísticos como se muestran en la Tabla 4.

5.3. Funciones de membresía

Las funciones de membresía son el núcleo de la lógica difusa, asignando valores cuantitativos a términos lingüísticos mediante grados de pertenencia, en contextos donde se prioriza la simplicidad y la claridad interpretativa, las funciones triangulares y trapezoidales son preferibles debido a su estructura lineal por partes y facilidad de cálculo [21]. En esta investigación, se optó por utilizar funciones de membresía triangulares para las variables de entrada, así como triangular y trapezoidal para la salida, coherentes con la necesidad de una modelación y computación eficiente y comprensible [19]. La Fig 5, exhibe una generalización de las funciones de membresía triangulares correspondientes a las variables de entrada las cuales fueron etiquetadas como se describe a continuación:

- 'NivelDeEntregaFrecuente'. Dichas funciones están categorizadas de la manera siguiente: 'BNE' representa un Bajo Nivel de Entrega, 'MNE' indica un Mediano Nivel de Entrega y 'ANE' denota un Alto Nivel de Entrega,
- 'NivelDeRefactoring'. Dichas funciones están categorizadas de la manera siguiente: 'BNR' representa un Bajo Nivel de Refactoring, 'MNR' indica un Mediano Nivel de Refactoring y 'ANR' denota un Alto Nivel de Refactoring.
- 'NivelDeIntegraciónContinua'. Dichas funciones están categorizadas de la manera siguiente: 'BNIC' representa un Bajo Nivel de Integración Continua, 'MNIC' indica un Mediano Nivel de Integración Continua y 'ANIC' denota un Alto Nivel de Integración Continua.
- 'NivelDeTesting'. Dichas funciones están categorizadas de la manera siguiente: 'BNT' representa un Bajo Nivel de Testing, 'MNT' indica un Mediano Nivel de Testing y 'ANT' denota un Alto Nivel de Testing.

La Fig. 6, exhibe las funciones de membresía correspondientes a la variable 'NivelDeAgilidad'. Dichas funciones están categorizadas de la manera siguiente: 'BNA' representa un Bajo Nivel de Agilidad, 'MNA' indica un Mediano Nivel de Agilidad y 'ANA' denota un Alto Nivel de Agilidad. El uso de la función trapezoidal es para dar mayor pertenencia a NAM. En la construcción del sistema difuso, se integró el nivel de

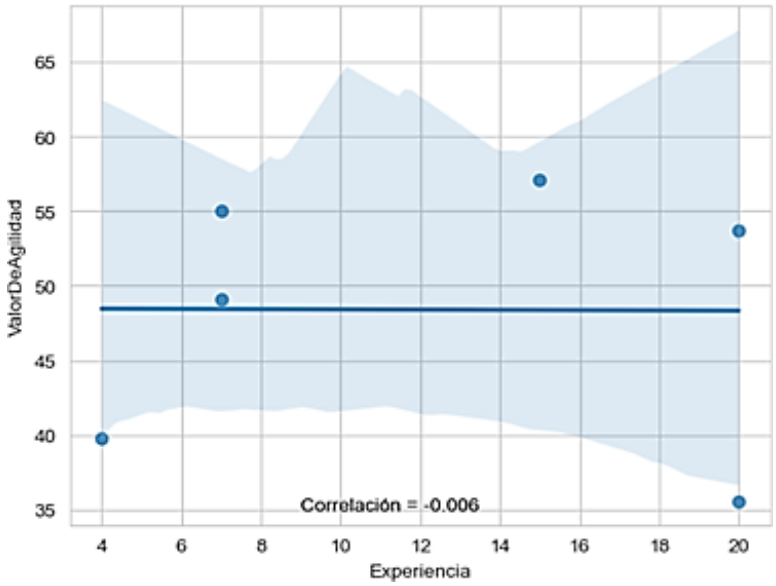

Fig. 5 Correlación de años de experiencia y nivel de agilidad.

agilidad dentro de las reglas If-Then, dichas reglas constituyen la estructura de las relaciones causales en sistemas de lógica difusa.

Estas reglas articulan los vínculos entre variables difusas, delineando cómo las condiciones (antecedentes) se mapean a las consecuencias (consecuentes) a través de la inferencia difusa. La utilización de estas reglas proporciona un mecanismo para deducir salidas difusas a partir de datos de entrada difusos, reflejando la complejidad y la naturaleza graduada de los procesos del mundo real [19].

5.4. Relevancia y ponderación de las actividades ágiles

Se calculó la relevancia de cada actividad ágil basándose en su porcentaje de importancia, tal como se detalla en la Tabla 4.

Este cálculo se efectuó mediante la aplicación de la fórmula **¡Error! No se encuentra el origen de la referencia.**, la cual fue diseñada para ponderar adecuadamente la contribución de cada actividad al nivel general de agilidad, las siglas se encuentran definidas en la Tabla 3. La formulación de las reglas *If-Then* refleja esta ponderación, permitiendo así que el sistema difuso evalúe con precisión el nivel de agilidad en el contexto del proyecto de software en estudio:

$$\text{Nivel De Agilidad} = NEF \times 0.38 + NR \times 0.39 + NIC \times 0.15 + NT \times 0.08. \quad (3)$$

Se desarrolló un sistema de inferencia difuso utilizando Python y el módulo Scikit-Fuzzy [22], una biblioteca especializada en algoritmos para lógica difusa. Se integraron y programaron las 81 reglas definidas, permitiendo una ejecución precisa del modelo de evaluación.

5.5. Instrumento de medición de las variables de entrada

Para obtener los valores de entrada del Sistema Difuso, se diseñó un cuestionario que fue contestado por los líderes de proyectos de desarrollo. Este instrumento incluye 10 preguntas para cada variable de entrada, las cuales son 'NivelDeEntregaFrecuente', 'NivelDeRefactoring', 'NivelDeIntegraciónContinua' y 'NivelDeTesting', cada pregunta con tres opciones de respuesta que reflejan la intensidad o frecuencia de las prácticas ágiles: '3' para un bajo nivel de implementación, '5' para un nivel medio, y '10' para un nivel alto. Las respuestas se suman, resultando en un puntaje total que conforma la entrada del sistema difuso.

6. Resultados y discusión

La aplicación de la metodología a casos de estudio reales revela su eficacia en capturar niveles de agilidad. La Tabla 5, muestra los resultados de la metodología aplicada a 6 empresas que se dedican al desarrollo de aplicaciones y utilizan metodologías ágiles.

De igual manera, las Fig. 7 y 8, representan de forma gráfica el resultado de la medición de la agilidad de las empresas E001, E002, E003, E004, E005 y E006 respectivamente.

En la representación gráfica de la medición de agilidad de las figuras Fig. 7 y Fig. 8, se destaca una barra vertical negra que indica el nivel de agilidad obtenido, evaluado en una escala de 0 a 100. Este valor se calcula utilizando el método del centroide para las áreas bajo las funciones de membresía.

Visualmente, a la izquierda de la gráfica, se presenta una función de membresía triangular para simbolizar baja agilidad; en el centro, una función trapezoidal denota agilidad media; y a la derecha, otra función triangular indica alta agilidad.

Además, el gráfico muestra una correspondencia cromática en la que el área coloreada aumenta conforme lo hace la relevancia del nivel de agilidad indicado. Esto permite una interpretación intuitiva de hasta qué punto el nivel calculado refleja una agilidad significativa en el contexto evaluado.

Además de las mediciones obtenidas, se incorporó en el instrumento de medición el registro de los años de experiencia de cada empresa evaluada. Este procedimiento se diseñó para investigar la posible correlación entre la experiencia acumulada de la empresa y su nivel de agilidad. Los datos se presentan detalladamente en la Tabla 6.

Aunque el conjunto de datos actual es limitado, se efectuó un análisis preliminar de correlación entre los años de experiencia de las empresas y sus niveles de agilidad medidos, resultando en un coeficiente de correlación de -0.006.

Este valor sugiere que no existe una relación significativa entre los años de experiencia y el nivel de agilidad alcanzado. La Figura 9, muestra el gráfico correspondiente. Se reconoce la necesidad de una base de datos más amplia para obtener resultados estadísticamente significativos.

7. Conclusiones

El modelo propuesto demuestra ser una herramienta prometedora para medir la agilidad en el desarrollo de software. Su capacidad para integrar la riqueza de la percepción humana con la rigurosidad de la evaluación cuantitativa aporta una comprensión más profunda de la agilidad práctica. Este enfoque ofrece un nuevo horizonte para investigaciones futuras y un marco para la mejora continua de las prácticas ágiles en la industria del software.

La adaptabilidad del sistema asegura que su aplicación puede extenderse a diversos entornos de desarrollo, haciendo de este modelo un candidato para la evaluación estándar de la agilidad en proyectos de software. Adicionalmente, este estudio no solo amplía el entendimiento actual sobre la medición de la agilidad, sino que también sienta las bases para el desarrollo de una metodología que permita evaluar el nivel de agilidad desde una perspectiva administrativa, excluyendo aspectos técnicos del desarrollo de software.

Este enfoque innovador posibilita la inclusión de organizaciones no dedicadas a este sector en la evaluación de agilidad de sus procesos administrativos. Al hacerlo, se amplía el alcance aplicable de las métricas de agilidad, facilitando su adopción en una variedad de contextos organizacionales.

Referencias

1. Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D.: Manifiesto por el desarrollo ágil de software http://agilemanifesto.org/iso/es/manifesto.html (2001)
2. Hohl, P., Klünder, J., van-Bennekum, A., Lockard, R., Gifford, J., Münch, J., Stupperich, M., Schneider, K.: Back to the Future: Origins and Directions of the "Agile Manifesto" – Views of the Originators. Journal of Software Engineering Research and Development, vol. 6, no. 1 (2018). DOI: 10.1186/s40411-018-0059-z.
3. Escobar-Sarmiento, V., Linares-Vasquez, M.: A Model for Measuring Agility in Small and Medium Software Development Enterprises. In: Proceedings of the XXXVIII Conferencia Latinoamericana en Informatica, pp. 1–10 (2012). DOI: 10.1109/clei.2012.6427226.
4. Royce, W.W.: Managing the Development of Large Software Systems: Concepts and Techniques. In: Proceedings of the 9th International Conference on Software Engineering, pp. 328–338 (1987)
5. Fox, A., Patterson, D.A.: Engineering Software as a Service: An Agile Approach using Cloud Computing. Strawberry Canyon LLC (2021)
6. Jacobson, I., Booch, G., Rumbaugh, J.: The Unified Software Development Process. Addison-Wesley Professional (1998)
7. Schwaber, K., Sutherland, J.: Scrum Guide V7 (2020)
8. Jeffries, R., Anderson, A., Hendrickson, C.: Extreme Programming Installed. Addison-Wesley Professional (2000)
9. Algarni, A., Magel, K.: Applying Software Design Metrics to Developer Story: A Supervised Machine Learning Analysis. In: IEEE First International Conference on Cognitive Machine Intelligence, pp. 156–159 (2019). DOI: 10.1109/cogmi48466. 2019.00030.
10. Rai, A.K., Agarwal, S., Kumar, A.: A Novel Approach for Agile Software Development Methodology Selection Using Fuzzy Inference System. In: International Conference on

Smart Systems and Inventive Technology, pp. 518–526 (2018). DOI: 10.1109/icssit.2018.8748767.
11. Meyer, B.: Agile: The Good, the Hype and the Ugly. Springer International Publishing (2014). DOI: 10.1007/978-3-319-05155-0.
12. Poppendieck, M., Poppendieck, T.: Lean Software Development: An Agile Toolkit: An Agile Toolkit. Addison Wesley (2003)
13. Cockburn, A.: Crystal Clear a Human-Powered Methodology for Small Teams. Addison-Wesley Professional (2004)
14. Beck, K., Andres, C.: Extreme Programming Explained: Embrace Change. Addison-Wesley (2004)
15. Ponce-Cruz, P.: Inteligencia Artificial con Aplicacion a la Ingeniería. Alfaomega (2010)
16. Voskoglou, M.: Fuzzy Sets, Fuzzy Logic and their Applications. MDPI Books (2020). DOI: 10.3390/books978-3-03928-521-1.
17. Williams, J.K.: Introduction to Fuzzy Logic. Massachusetts Institute of Technology, pp. 127–151 (2013). DOI: 10.1007/978-1-4020-9119-3_6.
18. Zadeh, L.: Fuzzy sets. Information and Control, vol. 8, no. 3, pp. 338–353 (1965). DOI: 10.1016/s0019-9958(65)90241-x.
19. Ross, T.J.: Fuzzy Logic with Engineering Applications. Wiley (2010). DOI: 10.1002/9781119994374.
20. Zadeh, L.A.: Outline of a New Approach to the Analysis of Complex Systems and Decision Processes. In: IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-3, no. 1, pp. 28–44 (1973). DOI: 10.1109/tsmc.1973.5408575.
21. Klir, G.J., Yuan, B.: Fuzzy Sets, and Fuzzy Logic: Theory and Applications. Pearson College Div (1995)
22. The Python Software Foundation: Scikit-Fuzzy. http://pypi.org/project/scikit-fuzzy/ (2024)